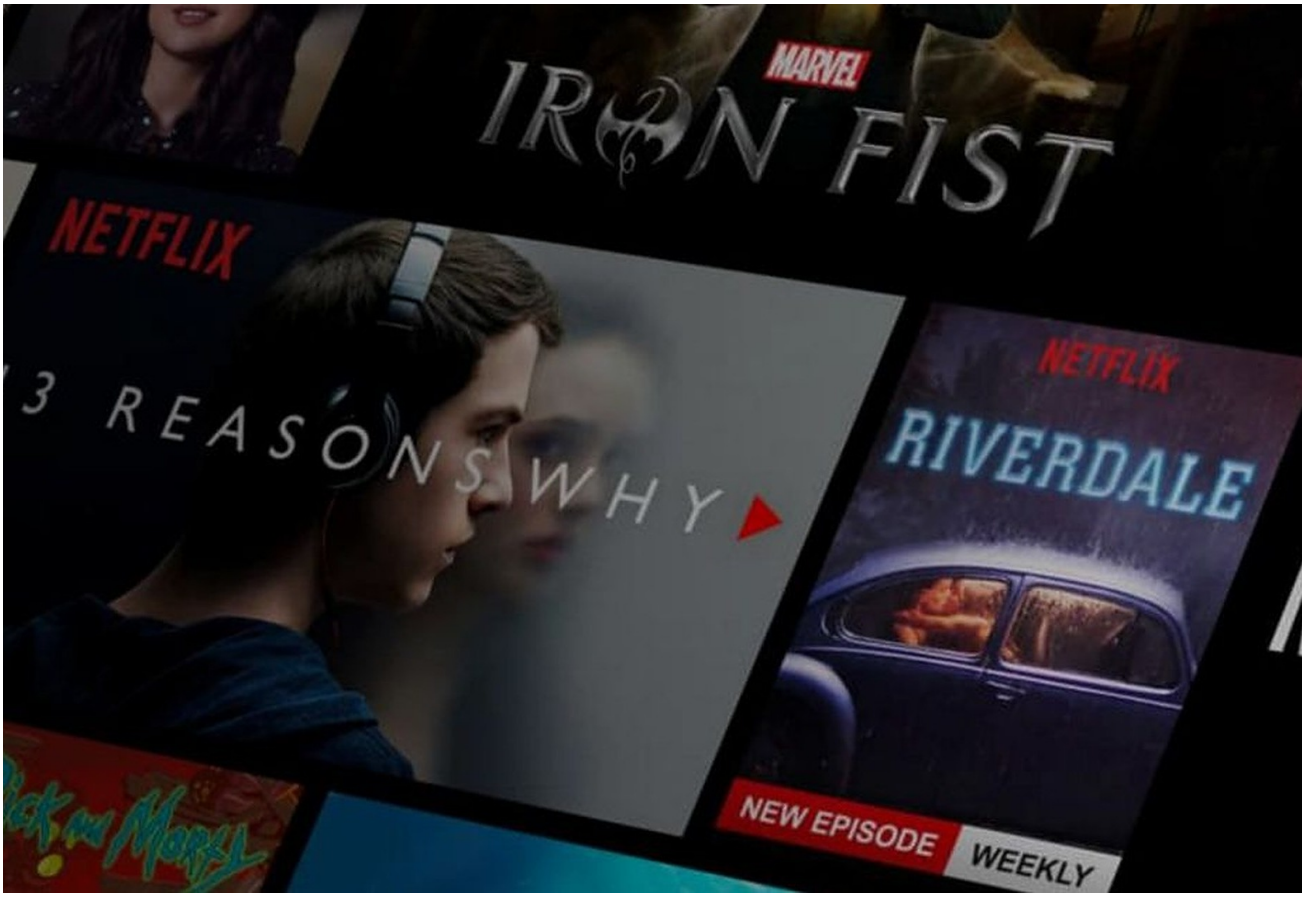
21 JULY 2023

MOVIE RECOMMENDATION SYSTEM USING SEARCH ALGORITHMS

AI PROJECT



Raghav Sharma (2020B2A42461H)
Shashank Gautam (2020B5A32378H)
Shresth Gattani (2020B1A72446H)

CRISP DM REPORT

BUSINESS UNDERSTANDING-

Objective:

This movie suggestion system's goal is to provide customers a list of films that are comparable to the movie they entered. To find and propose films that have traits, topics, or genres in common with the user's input movie, the system will make use of a variety of search techniques and similarity measurements.

Context:

There is a great quantity of material accessible to customers in the entertainment business, particularly in the movie sector. As a consequence, people often struggle to identify new films that suit their tastes or interests. By using data analysis methods to provide pertinent and individualised content recommendations, recommender systems play a critical part in alleviating this issue.

Source of the data:

This recommendation system's movie database is drawn from the "tmdb_5000_movies.csv" file. The dataset includes a selection of films together with their synopses, reviews, ratings, and other information. The algorithm will determine films that are comparable to the user's input by analysing this dataset, taking into consideration elements like title similarity, subject significance, and audience ratings.

Overview of the algorithm:

To find related films, the recommendation system uses a variety of search methods and similarity metrics:

1. Exact Match: When a user inputs a movie name, the system will seek for films whose names precisely match that input.

2. Partial Match: Films with titles that only partially match the user's input will be taken into account.

3. Levenshtein Distance: This method determines how similar two movie names are based on how many single-character modifications are necessary to transform one title into another.

4. Jaccard Similarity: This metric evaluates the intersection and union of word sets from movie titles to assess how similar two films are to one another.

5. Cosine Similarity: Using the cosine of the angle between the word frequency vectors of the titles of the films, this method determines how similar the films are to one another.

Benefits:

By making choices for movies that are appropriate for the user, encouraging movie discovery, and maybe raising user engagement on the platform, the movie recommendation system seeks to improve the user experience. The system may increase user happiness and retention by customising movie suggestions to each user's tastes.

Limitations:

Although the recommendation system makes use of a variety of search methods and similarity metrics, it is important to understand its limits. The approach largely uses movie titles and synopses to compare films, which may not account for all factors that contribute to a film's attractiveness. Furthermore, the accuracy and completeness of the underlying dataset have a significant impact on the efficacy of the suggestions. As a result, the method could not always take newer or less well-liked films into consideration.

Interaction with the user:

Users are prompted by the recommendation system to enter the title of a film they just saw and liked or are curious to learn more about. The system will create a list of related movies based on this input and show those titles, ratings, and summaries. After exploring these suggested films, users may choose which ones to view depending on their tastes.

DATA UNDERSTANDING-

Data Gathering:

- The 'tmdb_5000_movies.csv' CSV file contains the information that was utilised to generate this code. The CSV file includes details on movies, such as their names, vote averages, and summaries.

Description of the Data:

- The movie database is shown as a list of dictionaries, each of which includes details on a particular film. The following characteristics are relevant for the similarity search:
- "title"': The film's title.
- "vote_average": The film's overall rating.
- "Overview": A succinct synopsis or overview of the film.

Data Preprocessing: -

- Using the 'load_movies_db' function, the movie database is read from the CSV file and put into memory as a list of dictionaries.
- The titles are changed to lowercase in order to execute similarity searches efficiently and guarantee case-insensitive matching.

Data Exploration:-

- To locate related films, the code employs a number of similarity search algorithms:
- Exact Match: Films with the exact same input title.

- Partial Match: Films with the input in the title (case-insensitive search).
- Levenshtein Distance: Movies with the best Levenshtein distance match to the input.
- Jaccard Similarity: Films with names containing similar word groups.
- Cosine Similarity: Films with names that have comparable word frequencies.

Data Usage:

- A movie name input box is shown to the user.
- Based on the established search techniques, the code then identifies and presents the best related movies.
- The results provide the title of the film, the average number of votes, and a synopsis.

Data Restrictions:

- The similarity calculations are only based on movie names and synopses, which could not fully reflect all aspects of the content or genre of the films.
- The accuracy and applicability of the data in the movie database have a significant impact on how well similarity algorithms function.
- The method returns the top related movies using preset numbers (such as 6), which may not necessarily be the best option in all circumstances.

Data Challenges:

- It’s possible that the implementation didn't take into account details like synonyms or other movie titles.
- Some films may provide less accurate results if their names are brief or generic.

- The varied processing complexity of the selected similarity algorithms may affect the effectiveness of the search, particularly with a big movie library.

MODELLING-

1. Introduction to Modelling

In this part, we'll look at the many algorithms that are used to compare films based on the titles and input from the user. The code suggests films that closely like the user's input using a variety of search techniques, including exact match, partial match, Levenshtein distance, Jaccard similarity, and Cosine similarity. A short description of each algorithm and how it is used in the code will be given in this part.

2. Algorithm Descriptions

2.1. Exact Match:

- Description: This algorithm identifies movies whose titles exactly match the user's input.

- Implementation: The code performs a case-insensitive comparison between the user's input and each movie's title in the database to find exact matches.

2.2. Partial Match:

- Description: The algorithm identifies movies whose titles contain the user's input partially.

- Implementation: It searches for movie titles that have the user's input as a substring (case-insensitive).

2.3. Levenshtein Distance:

- Description: Levenshtein distance measures the similarity between two strings by calculating the minimum number of single-character edits (insertions, deletions, or substitutions) required to transform one string into another.

- - Implementation: The code computes Levenshtein distance between the user's input and each movie title using the SequenceMatcher class from the difflib module. It then selects the top 6 closest matches.

2.4. Jaccard Similarity:

- Description: Jaccard similarity measures the similarity between two sets by comparing the size of their intersection to the size of their union.

- Implementation: The code calculates Jaccard similarity between the set of words in the user's input and each movie title. The top 6 closest matches are chosen based on this similarity score.

2.5. Cosine Similarity:

- Description: Cosine similarity measures the cosine of the angle between two non-zero vectors in an inner product space.

- Implementation: The code computes the cosine similarity between the frequency vectors of common words in the user's input and each movie title. It then selects the top 6 closest matches.

3. Model Evaluation

The idea of model assessment doesn't apply in the usual sense since the main goal of this code is to provide suggestions depending on the user's input. However, by looking at the output produced by the various algorithms, we may assess the performance qualitatively. The quality, quantity, and variety of the movies in the database, as well as how well the search algorithms match the user's interests, all affect how successful the suggestions are.

4. Model Deployment

The model is implemented as a script so that users may interactively enter the title of the film. The script uses the developed search algorithms to find suggestions for

related movies after reading the database of movies from a CSV file. The user is presented with the findings in an intuitive way.

5. Conclusion

The numerous search strategies utilised in the code to discover comparable movie suggestions were provided in the modelling portion of the CRISP-DM report. Exact match, partial match, Levenshtein distance, Jaccard similarity, and Cosine similarity are a few of these methods. The model seeks to provide suggestions for films based on user input and several similarity criteria. The efficacy of the applied algorithms as well as the calibre and applicability of the movie database determine how successful the suggestions are. Users may interact with the model by entering the name of a movie, and the model will utilise the search algorithms to produce a list of related movies. Based on user comments and other data sources, the algorithms may be further enhanced and tuned.

EVALUATION-

To gauge the success of the recommendation system's efficacy and relevance, assessment is crucial. There are many assessment techniques that may be used:

a. In order to get input from actual users who engage with the movie similarity recommendation system, do user testing. User feedback may provide insightful information about the quality of the suggestions and how well they match users' interests.

b. Validation Dataset: Divide the movie database into a training dataset and a validation dataset in order to test the system's effectiveness. The training dataset was utilised to create the recommendation model. The accuracy and quality of the system's suggestions may be evaluated by contrasting them with the validation dataset.

c. Precision, Recall, and F1-Score: To assess the effectiveness of the recommendation system, compute precision, recall, and F1-score. F1-score combines both metrics to offer a fair assessment. Precision evaluates the proportion of relevant suggestions among all suggested films, recall measures the proportion of pertinent films discovered from the whole dataset.

DEPLOYMENT-

1. Introduction:

The CRISP-DM procedure ends with the deployment phase. The movie recommendation system created during the earlier stages (Business Understanding, Data Understanding, Data Preparation, Modelling, and Evaluation) is put into practise in this step. Based on different similarity criteria, including exact match, partial match, Levenshtein distance, Jaccard similarity, and Cosine similarity, the algorithm will assist users in finding related films.

2. Deployment Goals:

The deployment phase's main objective is to make the movie recommendation system accessible to end users and start making useful suggestions for them based on their movie interests.

3. Deployment Approach:

To deploy the movie recommendation system, we will follow these steps:

Step 1: Production Environment Setup:

Prepare the production environment, including the necessary hardware, software, and configurations, to host the movie recommendation system. Ensure that the environment is stable, secure, and can handle the anticipated user traffic.

Step 2: Integration with User Interface:

Integrate the movie recommendation system with a user-friendly interface. This could be a web application, mobile app, or any other user interaction medium. The users should be able to enter the movie name through the interface.

Step 3: Input Data Validation:

Implement input data validation mechanisms to ensure that the user-provided movie name is valid and free from potential errors.

Step 4: Calling Recommendation Function:

Upon receiving the user's input, call the 'get_similar_movies' function with the user-provided movie name and retrieve the similar movies from the deployed database.

Step 5: Displaying Results:

Display the list of recommended movies along with their ratings and overviews to the user through the user interface. Use text wrapping to format the movie overviews to improve readability.

Step 6: Error Handling:

Implement appropriate error handling mechanisms to address any potential issues that may arise during the recommendation process. Provide meaningful error messages to users when necessary.

Step 7: Load Balancing and Scaling:

Ensure that the deployed system can handle concurrent user requests effectively. Implement load balancing and scaling strategies to handle increasing user traffic.

Step 8: Monitoring and Maintenance:

Deploy monitoring tools to track the system's performance and identify any issues or bottlenecks. Regularly maintain and update the movie database to include new movies.

4. Deployment Results:

Users will be able to locate comparable films based on their preferences for films after the movie recommendation system has been implemented. A list of suggested films will be shown, along with reviews and ratings for each. This data may be used by users to find new films that interest them.

5. Conclusion:

The CRISP-DM process comes to a close with the implementation of the movie recommendation system. Users will get useful information from the system, which will aid them in picking films to view. The system will continue to be accurate and relevant with regular upkeep and upgrades.

SUMMARY-

Accomplishments:

1. Implemented movie recommendation system with search algorithms.
2. Allows users to find similar movies based on title similarities.
3. Efficiently combines results while removing duplicates.
4. Presents movie overviews in a readable format using "textwrap."

Potential Improvements:

1. Enhance user interface (web app or graphical interface).
2. Consider content-based recommendations (genre, actors, directors)
3. Implement collaborative filtering for user behavior-based suggestions.
4. Optimize algorithms and data structures for better performance.
5. Add sentiment analysis for understanding user preferences.
6. Explore machine learning integration for improved accuracy.
7. Augment data with external sources for richer movie information.
8. Include user feedback mechanism for better recommendations.
9. Implement robust error handling for smoother user experience.
10. Establish regular data updates for the movie database.